

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 03-288933

(43)Date of publication of application : 19.12.1991

(51)Int.Cl.

G06F 9/46

(21)Application number : 02-090314

(71)Applicant : HITACHI LTD

(22)Date of filing : 06.04.1990

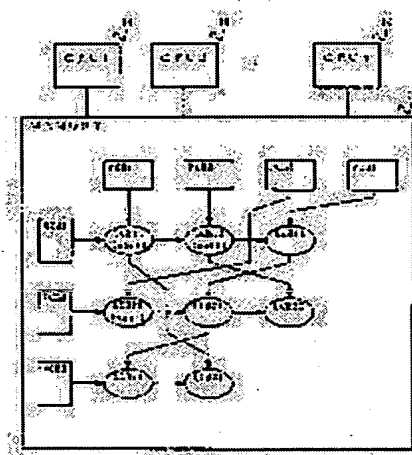
(72)Inventor : NITTA ATSUSHI
YONEDA SHIGERU

(54) METHOD FOR CONTROLLING ACCESS TO DATA RESOURCES

(57)Abstract:

PURPOSE: To obtain a high-functional locking mechanism with minimum spin weight by separately providing a locking word to show the possibility of access and a queue control word to control a queue showing the maintaining and waiting states of the lock, and utilizing a forming function in series at a low level.

CONSTITUTION: On a computer system where plural CPUs10 share a main storage device 20, plural process control blocks (PCB) 1 - PCB 4 execute locking requirements LCB 11 - LCB 34 to plural shared resources RCB 1 - RCB 3. In such a case, the locking word showing the state of the lock to the shared resource is separated from the queue control word to control the queue of the locking requirement and the access to the shared resource is efficiently formed in series among the plural processes. Thus, the high-grade locking mechanism can be realized without increasing the load of the CPU by the spin lock so much.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

⑩ 日本国特許庁(JP)

⑪ 特許出願公開

⑫ 公開特許公報(A) 平3-288933

⑬ Int. Cl.³
G 06 F 9/46

識別記号 庁内整理番号
3 4 0 H 8120-5B

⑭ 公開 平成3年(1991)12月19日

審査請求 未請求 請求項の数 6 (全15頁)

⑮ 発明の名称 データ資源に対するアクセスを制御する方法

⑯ 特 願 平2-90314

⑰ 出 願 平2(1990)4月6日

⑱ 発 明 者 新 田 淳 神奈川県川崎市麻生区王禅寺1099番地 株式会社日立製作
所システム開発研究所内

⑲ 発 明 者 米 田 茂 神奈川県川崎市麻生区王禅寺1099番地 株式会社日立製作
所システム開発研究所内

⑳ 出 願 人 株式会社日立製作所 東京都千代田区神田駿河台4丁目6番地

㉑ 代 理 人 弁理士 小川 勝男 外1名

明 細 書

1. 発明の名称

データ資源に対するアクセスを制御する方法

2. 特許請求の範囲

1. 多重プロセス、多重プログラミングシステムにおいて、複数のプロセスによって共有されるデータ資源へのアクセスを、各プロセスが発行するロック要求を許可するかまたは待たせることによって制御する方法であつて、共有資源およびロック要求に対応する制御ブロックを設け、該共有資源に対応する制御ブロックは、共有資源のロック状態を示すロックワードと、ロック要求に対応する制御ブロックを含むキューを管理するキューコントロールワードを持ち、該キューコントロールワードはある時点からある時点の間にキューへロック要求に対応する制御ブロックが追加されたことを検出する手段を含み、
(1) ロックワードを参照してロックが許可されるかどうかを判定し、次にロック要求に対応した制御ブロックをキューに追加し、もしロ

ック待ちが必要な場合はロックワードを再び参照して、ロック要求が許可可能な状態に変更されていたらロック待ちを回避するステップと、

- (2) キューを走査して許可可能なロック要求を見つけ、新たに許可されたロック要求に対応するようにロックワードを更新し、ロック待ちを解除し、キュー走査開始点からロック待ち解除時点の間に、ロック要求に対応する制御ブロックが、キューに追加されたことを検出したならば、キュー走査から処理を再試行するステップ、

を備えることを特徴としたデータ資源に対するアクセスを制御する方法。

2. 多重プロセス、多重プログラミングシステムにおいて、複数のプロセスによって共有されるデータ資源へのアクセスを、各プロセスが発行するロック要求を許可するかまたは待たせることによって制御する方法であつて、共有資源およびロック要求に対応する制御ブロックを設け、

該共有資源に対応する制御ブロックは、共有資源のロック状態を示すロックワードと、ロック要求に対応する制御ブロックを含むキューを管理するキューコントロールワードを持ち、該キューコントロールワードは最後にキューに入れられた制御ブロックを指すアンカポイントと、キューに制御ブロックが入れられた回数を記録する追加カウンタとを含み、ロック要求に対応する制御ブロックはキューに1つ先に入れられた制御ブロックを指すポイントと、要求しているロックの種別を示すワードと、ロック要求が待たされていることを示すウエイトフラグを含むことを特徴としたデータ資源に対するアクセスを制御する方法。

3. 請求項2においてさらに、

- (1) ロックワードを参照してロック要求が許可可能かどうかを判断するステップと、
- (2) ロック要求に対応する制御ブロックをキューに追加して、上記追加カウンタを増分するステップと、

ト中のロック待ちを解除するステップ、

を備えることを特徴としたデータ資源に対するアクセスを制御する方法。

5. 請求項2においてさらに、キューコントロールワードは、キューに対する専用アクセスの可能性を表す排他フラグと、キューに対する共用アクセスの可能性を表す共用カウンタを含み、ロック要求に対応する制御ブロックは、キュー中の制御ブロックが論理的に削除されたことを表す無効フラグを含み、

- (1) 排他フラグがセットされていなければ、共用カウンタを増分してキュー走査を開始し、排他フラグがセットされていればリセットされるのを待つステップと、
- (2) キュー走査を行って目的の制御ブロックに無効フラグをセットするステップと、
- (3) 共用カウンタを減じた結果が1以上の場合は共用カウンタを減じてキュー走査を終了し、0になる場合は共用カウンタを減じると同時に排他フラグをセットしてキューを走査し、

- (3) 上記キューへの制御ブロックの追加後、ロック待ちが必要であれば、ロックワードを再び参照して、ロックワード内容がロック許可可能に変更されていないかを確認し、もし許可可能であれば、ロック待ちの回避を行なうステップ、

を備えることを特徴としたデータ資源に対するアクセスを制御する方法。

4. 請求項3においてさらに、

- (1) キューコントロールワードの追加カウンタを過渡するステップと、
- (2) キューを走査してロック許可可能な制御ブロックのリストを作成するステップと、
- (3) 許可されるべきロックに応じてロックワードを変更するステップと、
- (4) 上記(1)において過渡した追加カウンタと現在の追加カウンタを比較し、もし異なっていれば上記(1)から処理を再試行するステップと、
- (5) 上記(2)で作成された制御ブロックのリス

無効フラグが設定された制御ブロックをキューから削除し、排他フラグをリセットしてキュー走査を終了するステップ、

を備えることを特徴としたデータ資源に対するアクセスを制御する方法。

6. 請求項1から請求項5のいずれかに示されたデータ資源に対するアクセスを制御する方法を備えたコンピュータシステム。

3. 発明の詳細な説明

〔産業上の利用分野〕

本発明は、データ資源に対するアクセスの制御方法、すなわち、電子計算機上のマルチプロセス、マルチプログラミング環境において、並行処理を行なうプロセス間でデータ資源を共有するためにアクセスを並列化する方法に関し、特に、複数の命令プロセッサが主記憶装置を共有するような密結合マルチプロセッサ環境に好適な、共有資源に対するアクセスの制御方法に関する。

〔従来の技術〕

電子計算機上のマルチプロセッサ、マルチプロ

グラミング環境では、多数のプロセスが様々な処理を並行して動作できる。プロセスは、オペレーティングシステムがCPUを割当てるための仕事の単位であり、その動作のためには、CPU、メモリ、I/O等の各種資源を必要とする。ここで、プロセスに固定的に割り付けられている資源の管理については、何等の問題も生じない。しかし、複数のプロセスによつて共有される資源に関しては、それらの資源に対するアクセスの競合を制御し、直列化する何らかの方法が必要である。

複数のプロセスによるアクセス競合を直列化する最も低レベルの方法は、一般にリード・モディファイ・ライト型命令と呼ばれる機械命令によつて与えられる。この命令は、主記憶上に格納された状態変数を参照し、その値にしたがつて状態変数を書き換える動作を、1つの原子的動作として実行する。例えば、HITAC Mシリーズ処理装置（日立製作所製電子計算機）におけるCS (Compare and Swap)命令、およびCDS (Compare Double and Swap)命令が、このような命令である。

呼ばれる資源対応の状態変数が設けられる。

ロック動作では、ロック要求プロセスは、ロックワードを参照して、もし使用中でなければロックワードを使用中に変更し、もし使用中であれば、再びロック動作を試行する（このロック動作の繰り返しはスピンウエイトと呼ばれる）。アンロック動作においては、プロセスは、ロックワードを使用中から非使用中に変更する。この方法では、ロックを確保しているプロセスは、共有資源に対して排他的な使用权を有し、他のプロセスはロック保有プロセスがアンロックを行なうまで、共有資源にアクセスできない。

ロック動作は、ロックワードを参照・更新するために、リード・モディファイ・ライト型命令を使用する。これは、あるプロセスが、ロックワードが使用中でないと判定してからロックワードを使用中に変更する処理の間に、他プロセスがロックワードを書き換えてしまうことを防ぐためである。

このようなロック機構は、しばしばスピントイ

これらの機械命令の詳細は、(株)日立製作所発行の「HITAC Mシリーズ処理装置」マニュアルに記載されている。

リード・モディファイ・ライト型の命令においては、その命令を実行するCPUが上記状態変数を参照してから更新するまでの間、その状態変数を含む主記憶領域をロックして、他CPUによる当該主記憶領域へのアクセスをウエイトさせるという制御を、ハードウェアによつて実現している。

ソフトウェアレイヤにおけるアクセスの直列化機構は、上記リード・モディファイ・ライト型命令を用いて実現される。マドニツク (S. E. Madnick) およびドノバン (J. J. Donovan) による「オペレーティング システムズ (Operating Systems)」(1974月刊, McGraw-Hill社)の4.5章に示されているように、このような機構は典型的には、「ロック」および「アンロック」と呼ばれる命令系列からなる。

この機構においては、ある共有資源が使用中であるかどうかを表示するために、ロックワードと

プのロックあるいは簡単にスピンロックと呼ばれる。

ソフトウェアによるロック機構には、上記スピンロックの他に、サスペンドタイプのロック（もしくはサスペンドロック）と呼ばれるものがある。サスペンドロックのロック動作においては、ロックが即時に確保できない場合、ロック要求プロセスは先入れ先出しのキューに入れられ、サスペンド状態となる。すなわち、ロック要求プロセスは、ロック待ちの間に、スピンウエイト処理によつてCPUを占有することなく、CPUを他の実行可能なプロセスに明け渡す。アンロック動作実行プロセスは、新たにロック確保が可能となるプロセスがキュー中に存在すれば、そのプロセスをキューから取り出してサスペンド状態を解除する。サスペンドロックは、スピンロックに比べて、CPUを有効に利用することが可能であるが、キュー操作とプロセスの状態変更処理の余分な動作を必要とする。また、一般にキューは、主記憶上に設けられたプロセス対応の制御ブロックをポインタチ

エーションで結ぶことによつて実現されるが、このポインタエーション自体が、ロック動作およびアンロック動作を並行して行なう複数のプロセスによつてアクセスされる共有資源であり、アクセスの直列化を必要とする。このキュー操作のアクセスの直列化は、スピンロックによつて実現されることが多かった。

特開昭60-128537には、スピンロックに依存することなく、リード・モディファイ・ライト型命令を直接使用して、サスペンドロックを実現する方法が示されている。この方法においては、ロックワードは、アクセスの可能性を示すロックフラグと、ロック待ちのプロセスを表現するキューのアンカとなるロックポインタからなり、ロックワード全体をリード・モディファイ・ライト型の命令で参照・更新することによつて、排他および共有の2つのモードを備えたロック機構を実現している。

〔発明が解決しようとする課題〕

上記の単純なスピンロック方法では、ロック確

保要求の競合が発生した場合、ロックを確保できた、ただ1つのプロセスを実行するCPUを除いては、CPUがスピンウエイト処理に使用され、本来の業務処理が中断される。これは、特に、複数の命令プロセッサが主記憶装置を共用するような、いわゆる密結合マルチプロセッサ環境においては、無視できない業務処理性能の低下を引き起こす。スピンロックは、オペレーティングシステムの最も基本的で単純な処理の直列化に使われるべきものであり、より高レベルの複雑な処理の逐次化には向いていない。

サスペンドロック方法は、スピンロックより処理が複雑であるが、ロック待ちのプロセスがCPUを占有することがないため、高レベル処理の逐次化に適している。しかし、サスペンドロック処理要求が頻発するようになると、ロック待ちのキュー操作を逐次化するためのスピンロックによる性能劣化が顕著になってくる。

上記特開昭60-128537に開示された方法を用いると、スピンロックを用いることなく、共用およ

び排他の2つのロックモードを備えたサスペンドタイプのロックを実現でき、高レベル処理の逐次化を効率よく行なうことができる。しかし、単なる共用・排他の2モードのロックだけでは、ソフトウェア階層のより上位の処理における、複雑な逐次化機能の要求に対応できない。1つのロックワード内に、共有資源の占有状態を表示するロックフラグと、ロック待ちを管理するキューのアンカポインタを設け、ロックフラグとロックポインタをまとめてリード・モディファイ・ライト型命令によつて参照・更新することが、実現可能なロック機能に対する制限をもたらす。これは、ロックワードの長さに制約が存在するためである。

典型的なハードウェアで実現されているリード・モディファイ・ライト型命令は、倍語長以下の主記憶領域を操作するものであり、通常アンカポインタは1語を占有するため、ロックフラグは1語長以内で表現されなければならない。これは、多種多様なロックモードを表現する場合に、大きな制約となる。

またこの方法では、ロック待ちを管理するキューの操作において、キューの最後の1つまたは連続する複数の要素を、リード・モディファイ・ライト型命令を用いて、要素のネクストポインタを更新することにより削除している。この方法は、キュー要素の削除を行なうプロセスが、同時にはただ1つしか存在しないという前提のもとでだけ正しく動作する。共用・排他の2モードのロックでは、キュー要素の削除は、ロックモードの変わり目となるアンロック処理時、すなわち、排他ロック解放、または最後の共用ロックの解放時にだけ実行されるため、上記前提条件が満たされる。しかし、より複雑なロック機能では、このような前提条件は必ずしも満足されず、複数のアンロック処理実行プロセスがキューを同時に操作するタイミングが発生しうる。もちろん、この2モードロック方法を、内部的な制御ブロック操作の逐次化に利用することにより、より高機能なロック処理プログラムを作成することが可能である。しかし、低レベルの機能（すなわちリード・モディフ

アイ・ライト型命令)を直接利用して、高機能なロック方法を実現できるならば、その方が望ましい。

本発明は、以下のような背景のもとに考案されたものであり、その目的は、高レベルの機能を提供したソフトウェアロック機構を、できるだけ低レベルの直列化機能を利用して、効率的に実現することである。

【課題を解決するための手段】

本発明の目的は、アクセス可能性を示すロックワードと、ロックの保有および待ち状態を示すキューを管理するキューコントロールワードを分離して設け、キューコントロールワードは、キューへの要素の追加をカウントする追加カウンタと、最後にキューに入れられた要素を指すキューアンカポインタを含むようにして、ロックワードおよびキューコントロールワードを以下のように参照・更新することによって実現できる；

(1) ロック処理は、ロックワードを参照してロック確保が即時に可能か、それとも待たなければな

らないかを判断し、ロック要求をキューに登録し、もしロックを待つのであれば、キュー登録後、ロックワードを参照してロック確保が可能かどうか再度確認し、ロック確保可能であればアンロック処理の一部を実行してロック待ちを回避する。

(2) アンロック処理は、はずキューコントロールワードの追加カウンタを記憶し、ロック許可処理を実行し、追加カウンタの値を、先に記憶しておいたカウンタ値と比較して、もし変更されていたら、アンロック処理を再試行する。

(3) ロック許可処理は、キューを走査してロック許可が可能な待ち状態のロック要求を確保し、ロックワードを新しいロック状態に更新し、ロック待ちを解除する。

また、本発明の目的は、キューコントロールエリアに、キューアクセスに対する排他・共用ロックを示すキューロックフラグを設け、また、キュー要素に削除された要素であることを示す削除フラグを設け、以下のようにキュー操作を行なうことによって実現される；

(1) キューへの要素の追加は、キューアンカポインタへ、追加要素のアドレスを設定し、追加カウンタをカウントアップすることによって行なう。

(2) キューからの要素の削除では、キューに対する共用ロックを確保して、キューを走査し、削除すべき要素に削除フラグを設定して、キューに対する共用ロックを解放する。この時、キューに対する共用ロックを最後に解放するプロセスが、共用ロックの解放と同時にキューに対する排他ロックを確保し、削除フラグが設定されている要素を実際にキューから削除する。キューに対して排他ロックが設定されている場合、キューに対する共用ロック要求はスピンウエイトする。

【作用】

共有資源に対するロックの状態を示すロックワードと、ロック要求のキューを管理するキューコントロールワードを分離したことによって、複雑なロック機能を実現することが容易になる。すなわち、ロックワードおよびキューコントロールワードはそれぞれ、本発明が適用されるコンピュー

タシステムのハードウェアによって規定される。リード・モディファイ・ライト型命令が扱える最大のオペランド長まで占有できるようになる。

一方、ロックワードとキューコントロールワードを分離して、別々に更新するため、ロックワードの状態とキューの状態との整合性をとるための機構が必要となる。ロック処理時の、キューイング後のロックワード再確認と、アンロック処理時の、キューコントロールワード中の追加カウンタの記憶および確認が、このような機構を与える。ロック処理時のロックワード再確認は、ロックワードを参照してから、キューイングするまでの間に、アンロック処理によって、ロックワードが変更されたことを検出する。また、アンロック処理時の追加カウンタの操作は、キューを走査してロック許可可能なロック要求を確定している間に、新たな待ち状態のロック要求がキューに追加されたことを検出する。これによって、ロック処理とアンロック処理が競合して、ロック許可可能であるにもかかわらず、待ち状態が解除されないロ

ク要求が発生してしまうタイミングを救済できる。

一方、キューコントロールワードの排他・共用ロック方法は、非常に少ないスピンウェイトで、一般的なキュー操作を行なうことを可能とする。キューへの要素の追加は、キューコントロールワードを、直接リード・モディファイ・ライト型命令で操作することによって実行され、キュー要素への削除フラグの設定は、キューに対する共用ロックを確保して実行される。これらの処理は、複数のプロセスによって同時に実行可能である。特に、アンロック処理で複数のプロセスがロック許可処理のためにキューを同時にアクセスすることが可能となる。共用ロックを最後に解放するプロセスは、共用ロックの解放と同時に、キューに対する排他ロックを確保し、キュー中の排他フラグが設定されている要素をキューから実際にはずして、排他ロックを解放する。この、排他ロック占有の間に、他プロセスがアンロック処理を実行しようとして、キューをアクセスする場合はスピンウェイトするが、このような競合が発生する確率

は十分に低いことが期待できる。

以上のように、本発明によれば、最小限のスピンウェイトの上に、高機能なロック機構を実現することができる。

〔実施例〕

以下、本発明の実施例を図に基づいて詳細に説明する。ここでは、2つの実施例を述べる。第1の実施例においては、全てのロック要求、すなわち、許可されているロック要求と、待たされているロック要求の双方が、キューに保持される。第2の実施例においては、待たされているロック要求だけで、キューに保持される。

(1) 第1の実施例

(1-1) コントロールブロックの構成

第1図は、本発明の第1の実施例の概要を示すコンピュータシステムのブロック図である。第1図では、プロセスが共有資源に対してロックを要求して許可されている。または待たされている、状態が、主記憶（もしくは仮想記憶）上の制御ブロック群と、それらの間のポインタチェーンによ

って表現されている。共有資源はリソースコントロールブロック(RCB)100によつて、プロセスはプロセスコントロールブロック(PCB)110によつて、ロック要求はロックコントロールブロック(LCB)120によつて、それぞれ示される。共有資源100に対するロック要求120は、ポインタチェーン140によつて、最新に発せられたロック要求から順につながれる。また、プロセス110が行ったロック要求120は、ポインタチェーン150によつてつながれる。すなわち、ロック要求120が、どのプロセスからの、どの共有資源に対する要求であるかということが、2種類のポインタチェーン140および150によつて一意的に特定される。

第2図に、本発明が実施されるコンピュータシステムの環境と、制御ブロック間のポインタチェーンの実例を示す。第2図では、複数のCPU10(CPU1~CPU_n)が、主記憶装置20を共有しているコンピュータシステム上で、複数のプロセスPCB1~PCB4が、複数の共有資

源RCB~RCB3に対してロック要求LCB11~LCB34を行つている様子が示されている。本発明は、このような、複数のCPUが主記憶装置を共有する密結合マルチプロセッサシステムでのマルチプログラミング環境において、複数のプロセス間の共有資源へのアクセスを効率的に直列化する方法を与える。本発明は、また、CPUが1台であるようなコンピュータシステムでのマルチプログラミング環境において、複数プロセスの共有資源へのアクセスを直列化する場合にも、有効である。

第2図に示すポインタチェーンのうち、横方向のチェーンは、特定の共有資源に対するロック要求のキューを表す。例えば、RCB1からLCB11, LCB12, LCB14に延びるポインタチェーンは、RCB1によつて代表される共有資源に対するロック要求を、要求発生時間の新しい順につなぐものである。LCB中に(wait)表示のあるものは、そのロック要求が現在待ち状態であることを示す。第2図では、RCB1に対して

最も古くに発行されたロック要求LCB14は許可されたが、その後に発行されたロック要求LCB12とLCB11は、即時にロックが許可されず、ロック要求LCB14が解放されるのを待っている状態が示されている。RCB2およびRCB3から右方向に延びるポインタチェーンについても同様である。

これに対し、縦方向のポインタチェーンは、特定のプロセスが発行したロック要求のリストを表す。例えば、PCB1からLCB11、LCB31へと延びるポインタチェーンは、PCB1によつて示されるプロセスが、ロック要求LCB11、LCB31を、これとは逆順に発行した事を示す。ロック要求31は許可されたが、ロック要求11は許可されず、プロセスPCB1は、現在待ち状態である。PCB2、PCB3、PCB4から下方向に延びるポインタチェーンも同様である。1つのプロセスは、0以上の任意の数だけ許可されたロック要求を持つことができるが、待ちのロック要求は高々1つしか持てない。

入れられるごとにカウントアップされ、ロック処理とアンロック処理の競合が発生したことを検出するために用いられる。キューアンカポインタANCは、ロック要求のキューを表すポインタチェーン140の始点である。

プロセスを代表するプロセスコントロールブロック(PCB)110は、そのプロセスに対して許可されている、または、許可されるのを待っているロック要求のリストを表すポインタチェーン150の始点となるポインタLPTを含む。

ロック要求を表すロックコントロールブロック(LCB)120は、RCBからのポインタチェーン140における次のLCBを示すポインタNIQと、PCBからのポインタチェーン150における次のLCBを示すポインタNISと、要求しているロックのモードを示すMODEと、フラグFLGを含む。フラグFLGは、LCBの状態を示すものであり、ロック要求が待ち状態であることを意味するWフラグ、LCBがキューすなわちRCBからのポインタチェーン140内で、

ここで再び第1図にもどつて、各制御ブロックの詳細を説明する。

共有資源を代表するリソースコントロールブロック(RCB)100は、共有資源に対するロックの状態を示すロックワードLWと、共有資源に対するロック要求のキューを管理するキューコントロールワードQCWを含む。ここで、本発明の1つの特徴は、LWとQCWが隣接している必要がないことである。ロックワードLWの内容の詳細については、後述する。キューコントロールワードQCWは、排他フラグX、共有カウンタSHR、削除フラグD、追加カウンタADD、キューアンカポインタANCからなる。排他フラグXと共用カウンタSHRは、複数プロセスによつて同時にアンロック処理が行われる場合の、キューに対するアクセスの直列化を制御するために用いられる。削除フラグDは、キュー中に、ポインタチェーンにはつながっているが、論理的には削除されているLCBが存在することを示すために用いられる。追加カウンタADDは、LCBが新たにキューに

無効であることを意味するIフラグ、ロック要求が許可可能となつて待ち状態を解除する必要があることを意味するPフラグを含む。LCBは、また、待ち状態を解除する必要があるロック要求をつなぐポインタチェーン160における次のLCBを示すポインタNIPを含む。

ロック処理/アンロック処理は、以上のコントロールブロックの他に、いくつかのワークレジスタ130を使用する。ワークレジスタの1つは、待ち状態解除を行なうべきロック要求をつなぐポインタチェーン160の始点となるポインタPPTとして使われる。もう一つのワークレジスタは、アンロック処理中で、新たに許可可能となつたロック要求のモードを記憶するためのAMODEとして使われる。

第3図は、ロックワードLWの内容を示している。本実施例におけるロックは、EU/ER/PU/PR/SU/SRの6種類のモードを持つ。

第4図は、これらのロックモード間の両立性を示す。第4図のマトリクス中で○印は2つのモード

が同時に許可可能であることを、×印は、2つのモードが相反であることをそれぞれ示している。このマトリクスからわかるように、EU/ER/PUモードのロックは、ただ1つのプロセスにだけ許可され、PR/SU/SRモードのロックは同時に複数のプロセスに許可される。このため、第3図のロックワードLW中では、EU/ER/PUモードは、各々フラグによつて、PR/SU/SRモードは各々カウンタによつて表現される。ロックワードLWは、また、許可を待っているロック要求の数を示すカウンタWAITを含む。1つでも待ちのロック要求が存在する場合、その後に来着したロック要求は全て待ちとなる。これは、ロック要求の時間順序性を保証するためである。ロックモードの種類、およびロックモードのロックワードLW中における表現方法については、第3図および第4図に示したものに限らず、任意のものが可能である。ここでのただ1つの制限は、ロックワードLWが、リード・モディファイ・ライト型の命令によつて、更新可能であるということ

CPUが主記憶を共有する密結合マルチプロセッサでは、あるCPUが実行中のプロセスが主記憶上の状態変数の内容を比較し、その値に従つてその状態変数を変更しようとする場合に、通常のロード/コンペア/ストア命令を使用したのでは、状態変数を読み込んで(ロード)、比較して(コンペア)、書き換える(ストア)処理の間に、状態変数が他のCPUによつて書き換えられ、処理の整合性が失われてしまうことがありうる。このためCS命令やCDS命令を使用して、比較と書き換えを原子的な1動作で行うようにするのである。

以下で、ロック処理/アンロック処理を、流れ図に基づいて詳細に説明する。なお、説明が不必要に複雑になることを避けるため、図では細かな初期設定処理や例外処理を省いてある。

(1-3) ロック処理

第5図は、ロック処理の流れ図である。ロックを要求するプロセスは、入力情報として、RCBアドレス、PCBアドレス、要求ロックモードを指定して、ロック処理をコールする。ロック処理

とである。

(1-2) CSおよびCDS命令

ロック処理/アンロック処理の詳細に入る前に、そこで使用されるリード・モディファイ・ライト型機械命令であるCS(Compare and Swap)命令とCDS(Compare Double and Swap)命令について簡単に説明しておく。

CS命令は、2つの汎用レジスタと主記憶(仮想記憶)上の1語長の領域をオペランドとし、第1のレジスタと主記憶(仮想記憶)上の1語の内容を比較して、もし両者が等しければ、その1語を第2のレジスタの内容で書き換えて条件コードを0に設定し、もし両者が異なれば、第1のレジスタにその1語の内容をロードして条件コードを1に設定する動作を、原子的な1動作として実行する、リード・モディファイ・ライト型の命令である。

CDS命令も、CS命令と同様の動作を実行するが、こちらは、4つの汎用レジスタを用いて倍語長の領域を操作するところが異なる。複数の

では、まずLCBを作成して、その内容を初期化し、PCBからのポインタチェーンにつなぐ(ステップ1001)。この初期化では、NIQおよびNIPは空値に、フラグは全てリセット状態に、MODEは要求ロックモードに設定される。また、LCBをPCBからのポインタチェーンにつなぐため、NISには旧LPTの値が、新LPTとしてLCBアドレスが、それぞれセットされる。

次に、ロックワードLWを参照して、要求したロックが許可可能か、待たなければならないかを判定し(ステップ1002)、CDS命令を用いて新しいロックワードをセットする(ステップ1003)。待ち状態のロック要求が存在せず、かつ、要求したロックモードが既に他のプロセスに許可されているロックモードと両立可能な場合にだけロックが許可される。新たなロックワードは、ロック許可可能な場合は、要求ロックモードに対応するフラグをセットする、もしくはカウンタを上げることにより、ロック待ちの場合は、WAITカウンタを上げることによつて得られる。

ステップ1003のCDS命令が失敗するのは、他プロセスのロック処理（ステップ1003）もしくはアンロック処理（後述の第7回ステップ1207）によるロックワードLWの更新とのアクセス競合が発生したためであり、ステップ1002から再試行する（ステップ1004）。ロック待ちが必要であればLCBのWフラグをセットする（ステップ1005、ステップ1006）。

続いて、CDS命令を用いて、LCBをキューに入れる（ステップ1007）。この処理は、現在のANCの値をNIQにコピーした後、CDS命令を用いて、QCWの内容が他プロセスによって変更されていないことを確かめながら、ANCの値をLCBアドレスに書き換え、かつADDカウンタの値を1だけ増加させることによって実行される。ADDカウンタが最大値である場合には、1ヘラツプアラウンドする。ステップ1007のCDS命令が失敗するのは、他プロセスのロック処理（ステップ1007）またはアンロック処理（後述の第6回ステップ1107、第8回ステッ

プ1302、1304）によるQCWの更新と競合が発生したためであり、キューイング処理を再試行する（ステップ1008）。

LCBをキューに入れた後、ロックが許可されるならばロック処理コール元にリターンし、ロック待ちが必要ならばモードの再確認を行なう（ステップ1009）。モード再確認処理では、再びロックワードを参照してロックが許可可能になっているかどうか確かめ（ステップ1010、ステップ1011）、ロック許可がやはり不可能である場合はロック要求プロセスを待ち状態にする

（ステップ1012）。もし、ロックワードが許可可能に変更されている場合は、ロック待ちを回避するためにアンロック2処理をコールする（ステップ1013）。このようなモード再確認を行うのは、ステップ1003でロックワードをセットしてから、ステップ1007でLCBをキューにつなぐまでの間に、現在ロックを許可されているプロセスがアンロック処理を実行してロックワードLWが、ロック許可可能な状態に書き換えら

れた時、今回のロック要求が待ちのままでキューに残る事を防ぐためである。LCBがキューにつながっていれば、アンロック処理がキューを走査して待ちのLCBを見つけることができ、ロック待ちは解除される。ロック処理でロックワードを参照する以前に、アンロック処理でのロックワード更新が行われていれば、ロック要求は待ちにならない。

問題は、競合するアンロック処理がロックワードを更新する以前に、ロック処理でロックワードを参照し、かつ、ロック処理でLCBをキューにつなぐ以前にアンロック処理でキューを走査する、というタイミングで2つの処理が実行された場合である。この場合、アンロック処理によって、このLCBの待ちが解除されることは期待できないため、LCBのキュー登録後にロック処理内でモードを再確認し、必要であればアンロック処理を模擬するのである。

従来の方法では、ロックワードLWとキューコントロールワードQCWは、一体化されており、

1回のCDS命令で同時に更新されていたため、このようなタイミングの狭間が発生することはない。本発明では、ロックの自由度を増大するために、ロックワードLWとキューコントロールワードQCWを分離しており、そのために、モード再確認処理が必要となつたのである。

(1-4) アンロック処理

第6回～第9回に、アンロック処理の流れを示す。アンロック処理は、2つのエントリポイントを持つ。ロックを許可されたプロセスが、そのロックを解放する場合にコールするのがアンロック1エントリポイントである。これに対し、前述したロック処理のモード再確認処理（ステップ1013）においてコールされるのがアンロック2エントリポイントである。入力情報は、双方のエントリポイントともLCBアドレスである。

第6回は、アンロック処理の初期部の流れである。通常のアンロック処理、すなわちアンロック1エントリポイントから始まる処理では、まずワークレジスタAMODEに、ロック解放によつて

発生するロックワードLWの変更分、すなわちLCBのMODEの逆数をセットする(ステップ1101)。次にCS命令を用いてLCBのIフラグをセットする(ステップ1102)。このCS命令は、Iフラグのセットが成功するまで繰り返される(ステップ1103)。続いて、LCBをPCBからのポインタチェーンよりはずす(ステップ1104)。すなわち、このLCBがポインタチェーンの先頭であればLPTに、そうでなければこのLCBの1つ前のLCBのNISに、このLCBのNISの値をコピーする。以上がアンロック1エントリポイントに個有の前処理であり、以後の処理はアンロック2エントリポイントと共通である。アンロック2エントリポイントから処理が開始された場合、ワークレジスタΔMODEの初期値は0である。

次に、ロックワードLW、およびキューコントロールワードQCW内のADDカウンタの値を、空きのワードレジスタまたはワークエリアに退避し(ステップ1105)、Xフラグがセットされ

ているかを見る(ステップ1106)。Xフラグがセットされていることは、他プロセスがキューを占有してアクセスを行っていることを意味しており、Xフラグがリセットされるまでスピンウエイトする。Xフラグがリセットされていることを確認したら、CDS命令でキューコントロールワードQCWのSHRをカウントアップする(ステップ1107)。このCDS命令が成功した場合、アンロック処理実行プロセスは、キューに対する共有アクセス権限を得たことになる。CDS命令が失敗するのは、他プロセスによるロック処理またはアンロック処理とのQCW更新における競合が発生したためであり、ステップ1105から再試行する(ステップ1108)。ここまでするアンロック処理の第1段階である。

第7図は、許可が可能となったロック要求をキュー中で見けて、待ちを解除する処理の流れである。この処理は、キューコントロールワードQCVのANCポインタからポインタチェーンをたどり、最も古くに待ち状態となったLCBを捜すことか

ら始まる(ステップ1201)。すなわち、Wフラグがセットされ、かつ、IフラグとPフラグがリセットされているような、ポインタチェーン中の最も後ろのLCBを見つける。許可可能なロック要求がもはや存在しない場合、このような状態のLCBは見つからず、ループ終了処理に移る(ステップ1202)。

待ちを解除すべきLCBが見つかったならば、CS命令でLCBにPフラグをセットする(ステップ1203)。CS命令が成功すれば(ステップ1204)、ロックワードの差分情報をワークレジスタΔMODEに加算(ステップ1205)、LCBをワークレジスタPPTからのポインタチェーンにつなぐ(ステップ1206)。ΔMODEの変更では、WAITカウンタを1減じ、LCBが要求するロックモードに対応するフラグのセット、もしくはカウンタの増分を行なう。また、LCBのNIPに現在のPPTの値をコピーし、PPTにこのLCBのアドレスをセットすることによって、LCBをPPTからのポインタチェーンにつ

なぐことができる。ステップ1203のCS命令が失敗するのは、LCBのフラグFLGに対するアクセスの競合が起きた場合であり、ステップ1201から処理を再試行する。

ステップ1206までの処理が完了したら、待ちを解除すべき次のLCBを捜すため、ステップ1201にもどる。このようにして、ΔMODEには、ロックワードを変更するための差分情報が累積され、PPTからのポインタチェーンには、待ちを解除すべきLCBのリストが形成される。

LCBサーチのループから抜けると、ΔMODEに蓄積されたモード差分情報を用いて新しいロックワード内容を作り、CDS命令でロックワードLWにセットする(ステップ1207)。この場合、CDS命令でLWと比較するのは、ステップ1105で退避しておいたロックワードの内容である。このCDS命令が失敗するのは、ステップ1105からステップ1207までの間に他プロセスがLWを更新したためであり、再び、ループの先頭(ステップ1201)にもどって、待ちが

解除できるLCBがないか確かめる(ステップ1208)。CDS命令が成功したならば、PPTからのポインタチェーンにつながれているLCBの持ちを全て解除し(ステップ1209)、第8図のアンロック終了処理に制御を移す。

第8図は、アンロック処理の終了部の流れである。ここでは、ステップ1107で確保したキューに対する共用アクセス権限の解放処理を行なう。まず、QCWのSHRカウンタの内容を見て(ステップ1301)、キューにアクセスしている他プロセスが存在する、すなわちSHRの値が1よりも大きい場合は、CDS命令を用いてSHRの値を減じる(ステップ1302)。もし、アンロック処理1エントリポイントから処理が開始されているのであつたならば、このCDS命令で同時にDフラグをセットする。CDS命令が成功すればアンロック処理を終了してコールもとヘリターンする(ステップ1303)。一方、キューをアクセスしているプロセスが自プロセスだけの場合、SHRの値は1である。この場合は、CDS命令

を用いてSHRの値を減じると同時にXフラグをセットする(ステップ1304)。もし、アンロック処理1エントリポイントから処理が開始されている場合には、やはり、Dフラグを同時にセットする。ステップ1304のCDS命令が成功してQCWにXフラグをセットするということは、キューに対する排他アクセス権限を確保できたことを意味する。この場合、第9図のLCB削除処理に制御を移す(ステップ1305)。

ステップ1302もしくはステップ1304のCDS命令が失敗するのは、他プロセスによるロック処理またはアンロック処理とのQCWアクセスの競合が発生した場合である。どちらの処理と競合したかを確認するために、ADDの値が、ステップ1105で退避した値と等しいかどうかを見る(ステップ1306)。ADDの値が変わっていなければ、他プロセスのアンロック処理との競合であり、ステップ1301からQCWの更新処理を再試行すればよい。ADDの値が変わっていれば、他プロセスによるロック処理と競合が発

生したためであり、ステップ1201からキュー走査処理全体をもう一度行う必要がある。ロック処理によつてADDが更新されたということは、新たなLCBがキューに追加されたことを意味する。このLCBが、ロック許可可能な、待ち状態のLCBではないかどうかを確認するために、キュー走査を再試行するのである。このようなLCB状態が発生するのは、まずロック処理がロックワードLWを参照して待ちを決定し(ステップ1002)、次にアンロック処理がキューを走査して待ちを解除可能なLCBを捜し(ステップ1201～ステップ1206)、その後ロック処理がLCBをキューに追加してプロセスを待ち状態にし(ステップ1007～ステップ1012)、最後にアンロック処理が上記待ちロック要求を許可可能とするようなモードにロックワードを更新する(ステップ1207)、というタイミングで2つの処理が同時実行される場合である。ロック処理がLCBをキューに追加するよりも、アンロック処理がロックワードを更新する方が先であれば、ロック処

理のモード再確認処理(ステップ1010)が、競合発生を検出してLCBの持ちを解除する。このように、ロック処理におけるモード再確認動作と、アンロック処理におけるADDカウンタ確認動作の双方を備えることによつて、ロック要求が不要に待たされるタイミングが発生することを防止できる。

最後に、第9図にしたがつて、LCBのキューからの削除処理を説明する。まず、QCWのDフラグを参照して、キュー中に削除すべきLCBが存在するか判定する(ステップ1401)。Dフラグがセットされていないければ、Xフラグをリセットしてアンロック処理を終了する(ステップ1412)。Dフラグがセットされている場合には、ANCからポインタチェーンをたどつて最初のLCBを得る(ステップ1402)。最初のLCBのIフラグがセットされていれば、CS命令でそのLCBをキューからはずして(ステップ1404、ステップ1409)、LCBをメモリの空き領域として解放する(ステップ1411)。先頭の

LCBをキューからはずすのは、LCBのNIQの値をANCに設定することによって行う。ステップ1409のCS命令が失敗するのは、他プロセスのQCW更新処理との競合した場合であり、ステップ1402から処理を再試行する（ステップ1410）。ステップ1411でLCBを解放した場合も、次のLCBを得るためにステップ1402にもどる。このようにして、ANCの指す先頭のLCBが存在しなくなった場合は、ループを抜ける（ステップ1403）。

先頭のLCBのIフラグがセットされていない場合は、NIQをたどって次のLCBを得（ステップ1405）、そのLCBのIフラグがセットされていたら（ステップ1407）、LCBをキューからはずして解放する（ステップ1408）。この場合、1つ前のLCBのNIQに、はずすべきLCBのNIQをコピーすることによって、キューからの削除が実行される。ここでのNIQの更新にCS命令を用いる必要はない。このようにして、キューの最後のLCBまで処理が進んだら

ループから抜ける（ステップ1406）。

アンロック処理では、キュー中の任意の位置のLCBが削除および解放される可能性があるため、キューのポインタ操作を行なうプロセスを、ただ1つに限定する必要がある。LCBの削除処理はQCWのXフラグをセットして実行するため、他プロセスのアンロック処理によるキュー操作と競合してポインタチェーンが不正になることはない。他プロセスのアンロック処理は、Xフラグがリセットされるまでスピンウエイトする（ステップ1105～ステップ1106）。Xフラグは、SHRを1から0に変更するプロセス、すなわち、並行してキューアクセスを行っていた複数のプロセスのうちで最後にキューアクセスを終了するプロセスによってだけセットされ、IフラグがセットされたLCBのキューからの削除が終了と直ちにリセットされる。したがって、スピンウエイトが発生する確立は非常に小さいものである。一方、ロック処理によるキューアクセスの競合は、ステップ1409のCS命令で検出する。ロック処理

では、キューに対するアクセスがANCへの新LCBの追加に限定されているため、スピンウエイトを用いないでアンロック処理キュー操作と処理を直列化することができるのである。

(2) 第2の実施例

第10図は、本発明の第2の実施例の概要を示すコンピュータシステムのブロック図である。第2の実施例が第1の実施例と異なる点は、第2の実施例においては、LCBは、もはやあるプロセスがロックを許可されている状態を示す役目を持たずに、ロック待ち表すためだけに用いられることである。したがって、各プロセスには高々1個のLCBしか対応せず、PCB、およびPCBからLCBをつなぐポインタチェーンは不要となる。RCBやワークレジスタに関しては、第1の実施例と同様である。

第2の実施例におけるロック処理およびアンロック処理は、キューにLCBを登録するのが、ロック待ちの場合だけであることを除いては、第1の実施例とほぼ同様であり、ここで詳しく述べる

ことはしない。第5図において、ステップ1001を削除し、ステップ1005でYESの場合は即時にリターンし、ステップ1006でLCBを作成するように処理を変更すれば、第2の実施例におけるロック処理の流れ図が得られる。同様に、第6図において、全てのアンロック処理をアンロック2エントリポイントから開始するようにし、第7図において、ステップ1203でLCBのPフラグと同時にDフラグもセットするように処理の流れを変更すれば、第2の実施例におけるアンロック処理の流れ図が得られる。

〔発明の効果〕

以上のように、本発明によれば、複雑なモードを備えたロック機構を、リード・モディファイ・ライト型命令と最小限のスピンロックだけによって効率的に実現することができる。特に、複数のCPUが主記憶を共有するような、密結合マルチプロセッサ環境において、スピンロックによるCPUの負荷増大をあまり発生させずに、高度なロック機能を実現できる効果がある。

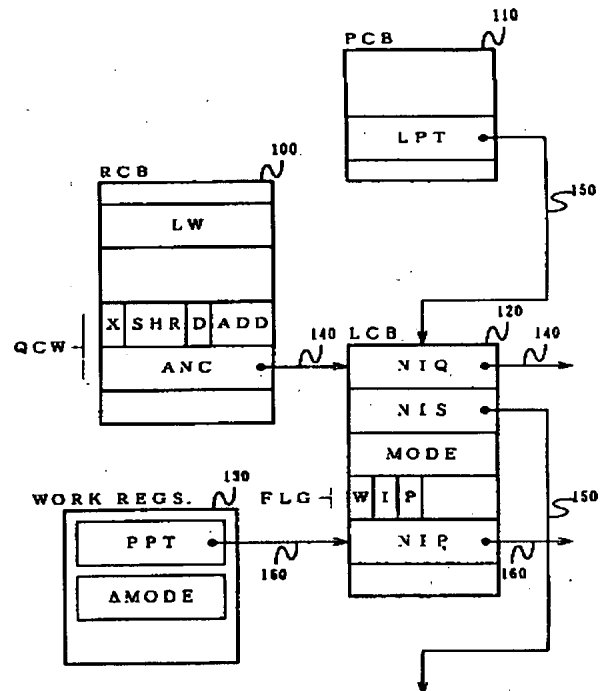
4. 図面の簡単な説明

第1図は本発明の第1の実施例を示すコンピュータシステムのブロック図、第2図は本発明が有効に働く計算機環境と主記憶内容を示したブロック図、第3図はロックワードの詳細の説明図、第4図はロックワードの両立性マトリクスの説明図、第5図はロック処理の流れ図、第6図から第9図はアンロック処理の流れ図、第10図は本発明の第2の実施例を示すコンピュータシステムのブロック図である。

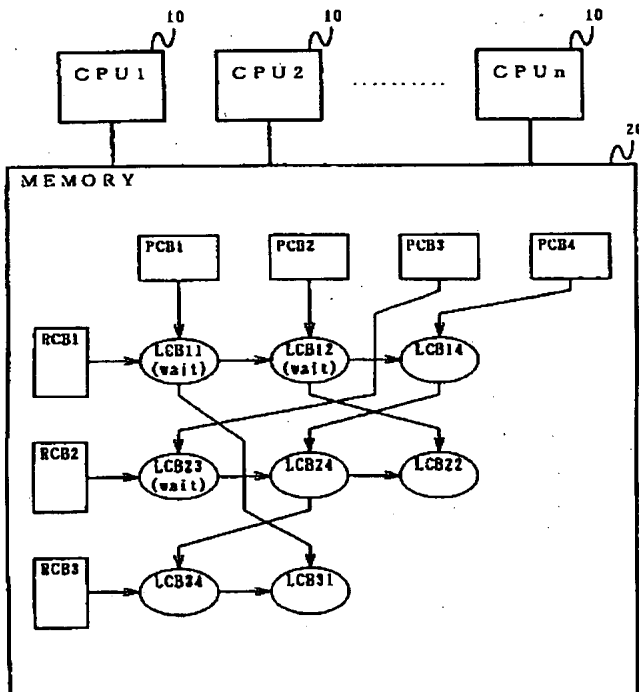
10…CPU、20…主記憶（仮想記憶）装置、
100…リソースコントロールブロック(RCB)、
110…プロセスコントロールブロック(PCB)、
120…ロックコントロールブロック(LCB)、
130…ワークレジスタ、140…ANCポイン
タチェーン、150…LPTポインタチェーン、
160…PPTポインタチェーン。

代理人 弁護士 小川勝男

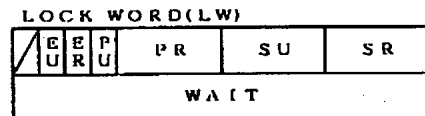
第 1 図



第 2 図



第 3 図



第4図

ロック両立性マトリクス

要求された ロックモード 保有された ロックモード	SR	SU	PR	PU	ER	EU
SR	O	O	O	O	X	X
SU	O	O	X	X	X	X
PR	O	X	O	X	X	X
PU	O	X	X	X	X	X
ER	X	X	X	X	X	X
EU	X	X	X	X	X	X

O : 両立する
X : 両立しない

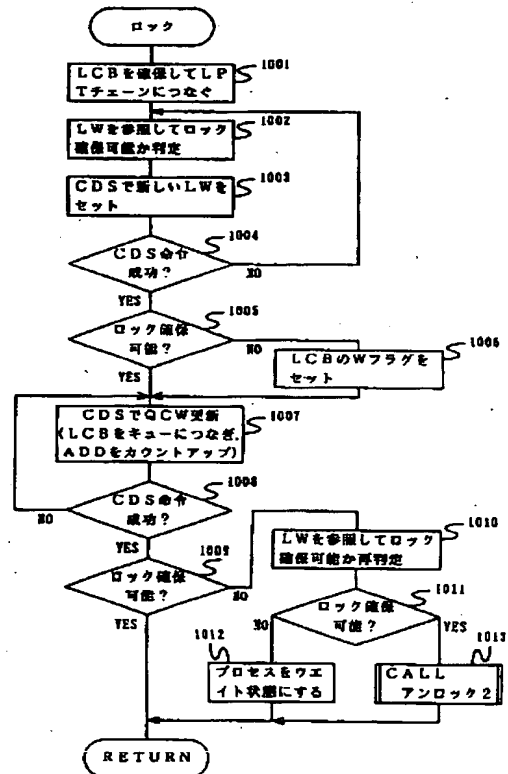
(1) モードの1文字目=他プロセスに許可する処理

S (Shared) : リード/ライトを許す
P (Protected) : リードは許すがライトは許さない
E (Exclusive) : リード/ライトを許さない

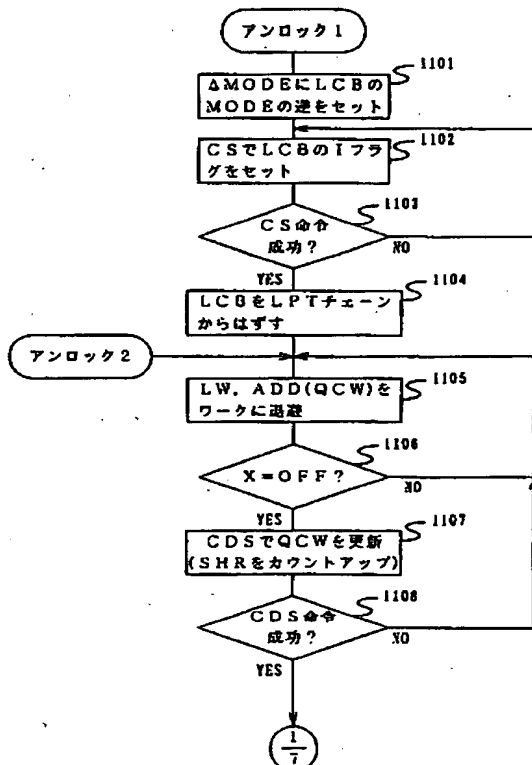
(2) モードの2文字目=自プロセスで行なう処理

R (Retrieve) : リードを行なう
U (Update) : リード/ライトを行なう

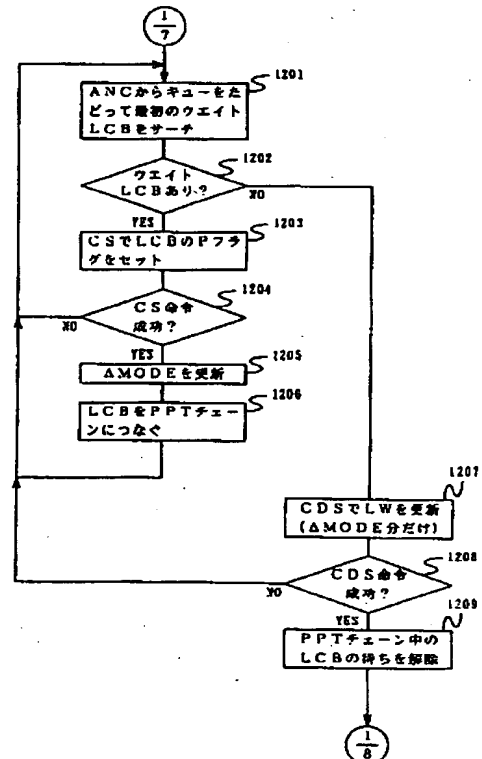
第5図



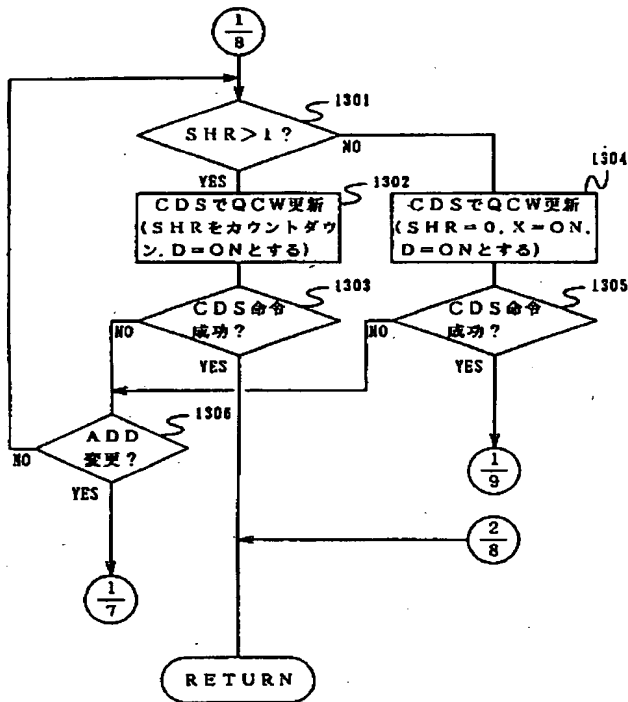
第8図



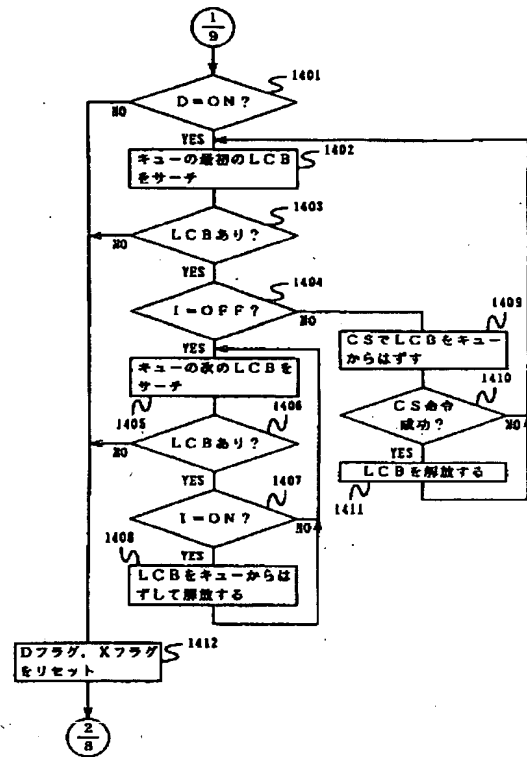
第7図



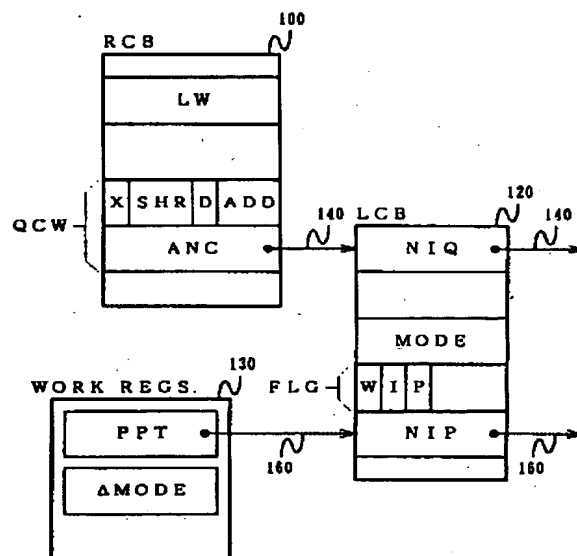
第8図



第9図



第10図



**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☒ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.